Victor Feight

CS-320

Module 5 Journal

* What were the software testing techniques that you employed for each of the milestones? Describe their characteristics using specific details.

Although *Task Service, Contact Service,* and *Appointment* service all had slightly different requirements, I was able to employ similar software testing techniques for each program. In particular, I used a Test-Driven Development approach wherein I extracted the necessary test cases by analyzing the user requirements for each module/service. After writing a list of test cases for a certain function, I would write the function and ensure all test cases passed. For example, I would write a test case to add a contact like such:

TaskService service = **new** TaskService();

// test adding first contact

*assertEquals*(**true**, service.addTask(task));

And afterward, I would write out the function to run against said test case like so:

**if**(!IDpresent) {

taskList.add(newTask);

System.***out***.println("Added task");

**return** **true**;

} **else** {

System.***out***.println("Duplicate ID: Failed to add task");

**return** **false**;

}

More specifically, I employed the technique of Input, as most of the input for these modules had to be checked before instantiating the object with bad values. I utilized @Test annotations for the JUnit testing, which allowed for both easier automation and some regression testing to catch any errors after changing a certain function.

* What are the other software testing techniques that you did not use for the milestones? Describe their characteristics using specific details.

Certain techniques that were not used include Integration testing, System testing, or any form of Black Box testing. Specifically, the tests that were used tend to be more white box in nature, as the developmental codebase was integrated with the unit testing. Black box techniques such as boundary value analysis and equivalence partitioning were not used. According to our readings in *Software Testing: An ISTQB-BCS Certified Tester Foundation,* Boundary value analysis involves testing at the boundaries of input, whereas equivalence partitioning involves dividing the input set into different partitions of testable data. The book describes Integration testing as integrating individual modules/functions with other modules/functions to ensure they are compatible in the system. System testing involves validating not only the software works, but that all of the specifications are met in terms of the whole physical computer hardware and software system.

* For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.

JUnit testing can be adopted in any incremental development environment and its uses involve catching bugs early by testing early. Similarly, regression testing is beneficial in catching bugs before they creep too late into the Software Development Life Cycle. Automation technique ensures these tests can be run on a regular basis to test modules. Integration testing can help ensure each of the modules is sending and inputing data correctly between each other. System testing can ensure each physical computer, OS, and browser is working as per specifications of the system. Any form of black box testing can complement white box testing to help test performance issues, or break the code at certain input boundaries without needing access to a codebase. This ensures that testing and development can be done in two separate, yet co-aligned environments.